Abstract

Java is a widely used programming language Programs written in Java are compiled to architecturally neutral bytecodes which can be executed on any system with a Java Virtual Machine (JVM). Thus, Java programs are portable, making the Java programming language a popular choice among programmers. In this thesis, we study the performance of a JVM

Our approach for gathering performance data makes use of the hardware performance counters implemented in recent microprocessors. We carried out our performance measurements using the kaffe JVM executing on a Pentium II microprocessor The Pentium II performance counters can be configured to count events such as number of instructions executed, instruction TLB misses, branch mispredictions, and cache misses We implemented a sampling based mechanism for gathering performance data of the kaffe JVM execution on the Linux operating system This mechanism obtains accurate performance data at a very low overhead – for a sampling rate of 50μs the time overhead is about 1%. Our measurements isolate the garbage collection phase behaviour, and show the impact of garbage collection on the locality of kaffe execution.

Just-In-Time (JIT) compilation is an alternative mechanism to interpretation for executing Java bytecodes We conducted a performance comparison of kaffe execution using interpretation and JIT, using metrics like number of native instructions executed, instruction and data cache miss rates, instruction TLB miss rates and branch misprediction ratio. Our discussion of the data identifies the reasons for performance differences observed in the interpreted and JIT compiled execution of bytecodes

We next looked at how to improve the code generated by the JIT code generator,

including (i) removing the function call overhead by in-lining code and improving the code generated for the LCMP bytecode, (ii) improved array index bounds checking, and (iii) code optimisation to reduce the number of mis-aligned code references

Finally, we compared the performance of the interpreted kaffe JVM running on two structurally different operating systems – Linux and the VSTa microkernel. We discuss the JVM implementation issues in this context, and present performance measurement data which suggest that compute intensive benchmark performance is not affected by the operating system structure, while for I/O intensive benchmarks, the performance of JVM on the microkernel is poorer. We identify the reasons for this and suggest possible solutions